Document Analysis Systems Development and Representation through the Object-Process Methodology

Dov Dori

Information Systems Engineering
Faculty of Industrial Engineering and Management
Technion, Israel Institute of Technology
Haifa 32000, Israel
dori@ie.technion.ac.il

Abstract. Object-Process Methodology (OPM) calls for integrated system development, supported by Object-Process CASE Tool – OPCAT – a Computer Aided Software Engineering tool, which supports development and standard representation of industrial systems. OPCAT is designed to provide an Integrated System Engineering Environment (ISEE). Due to their generic nature and comprehensiveness, OPM and OPCAT are suitable for both developing Document Analysis Systems (DAS's) and representing them in a clear and standard way. We present the principles of OPM and show how it can be used as an ISEE for generic DAS development.

1 Introduction

Traditional inputs, such as labor, land and capital, which have played a major role in economics until the last decade, are no longer the major catalysts for economic sustainability and growth. The emerging important resource for the enterprise is an intangible one – the knowledge stored in documents and inside the heads of individuals.

In the last few decades, information has become a prime resource, whose importance is constantly increasing. For lack of a better word, we use the term information to designate a hierarchy of concepts that are built on top of each other. Starting with raw data, stored in ever more powerful and sophisticated machines, from which information is constructed, the hierarchy proceeds to knowledge, which is the result of human and machine information processing, all the way up to expertise. Expertise is the trait of a person who has accumulated and assimilated knowledge to the degree that using natural intelligence, s/he can deduce and generate new knowledge that is useful in the domain. As we climb up this information hierarchy, the emphasis shifts from the power and sophistication of machines to those of human beings.

High technology societies rely heavily on this human resource of natural intelligence and innovativeness. As we proceed toward the 21st century, the quality of this resource

S.-W. Lee and Y. Nakano (Eds.): DAS'98, LNCS 1655, pp. 271-282, 1999.

depends more and more on information already present in the organization or in its environment. In order to succeed, companies today need to learn how best to find, capture and exploit this precious resource. The ability of an enterprise to develop and make use of Document Management Systems (DMS's) is becoming as crucial s ever. Such systems, of which document analysis systems are but one component, are aimed at that collecting, storing, analyzing and making use of raw data and information through intelligent processing mechanisms. The ultimate goal of such systems from the enterprise viewpoint is to generate useful knowledge that serves its critical success factors. This domain of research and development is emerging as a new area called Knowledge Management (KM), with DMS's providing the infrastructure for KM.
For most enterprises, the knowledge is there, but it is not readily and widely available. In most cases, it does not even have to be created. Rather, it has to be captured. Enterprises have not been capturing "what they know". KM technologies are designed to pave the way to solving this problem. By capturing and finding out what they know, and then disseminating this knowledge to the right people, enterprises will be able to streamline their operations and improve their business processes and customer relationships.
To cope with the challenges and high-level KM requirements, the development of DMS's and general and DAS's in particular, must rely on a sound, integrated systems engineering environment. The Object-Process Methodology (OPM) is proposed here as an ideal infrastructure for this endeavor.

2 Object-Process Methodology (OPM)

The Object-Process Methodology (OPM) [1], [2], [3], [4] is a system development approach that has been developed to cater to the natural train of thought humans normally apply while trying to understand and build complex systems. In such systems, it is usually the case that structure and behavior are intertwined so tightly, that any separation between them is bound to further complicate the already complex description. Founded on General Systems Theory, OPM is generic and therefore most suitable for specifying systems of virtually any domain. In particular, DAS's are prime candidates for making use of OPM as we show in this work.
OPM incorporates the static-structural and behavioral-procedural aspects of a system into a single unifying model. This approach is counters contemporary object-oriented systems development methods, such as the accepted UML standard [5], which requires no less than eight different models, each with its own . OPM achieves model integration by treating both objects and processes as equally important things (entities) in the system's specification.
In OPM, objects are viewed as persistent, state preserving entities that interact with each other through processes. Processes are transient, transforming entities that affect objects by changing their state, or by generating or consuming objects. This structure-behavior unification into a single model results in synergy of unparalleled expressive power. By incorporating analysis, design, implementation, and deployment within one frame of reference and enabling smooth transitions among theses phases, OPM provides an Integrated Systems Engineering Environment (I SEE).

OPM uses Object Process Diagrams (OPDs) for expressing the objects of a modeled system and the processes that affect them. Figure 1 shows three OPDs, which describe how processes affect objects. This figure, as well as the others in this work, was drawn using OPCAT Version 1.6, which is available from the URL http://iew3.technion.ac.il:8080/~dori/opcathp/index.htm. As Figure 1 shows, objects and processes in an OPD are denoted within rectangles and ellipses, respectively, while object states are marked as round-cornered rectangles.

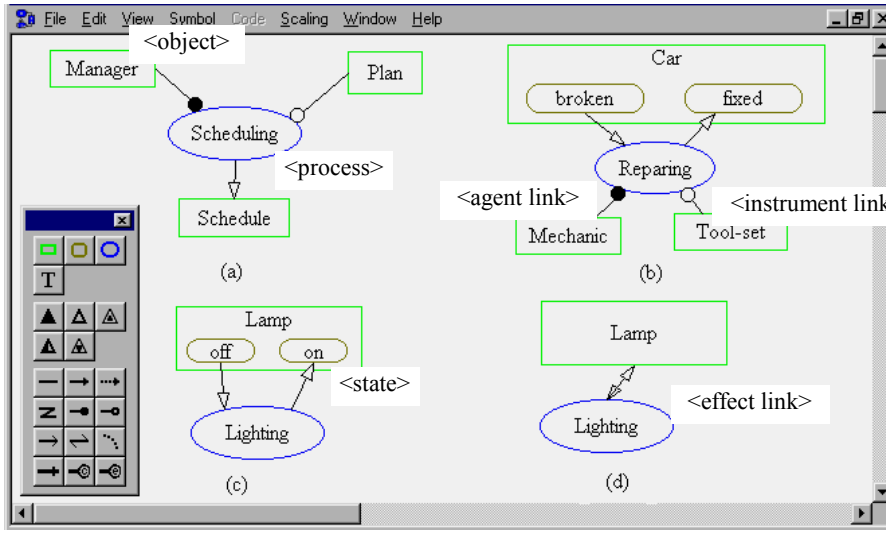

Fig. 1. OPDs showing how processes generate objects and change object states. (a) The object Manager is an agent that generates the object Schedule through the process of Scheduling using Plan as an instrument. (b) The agent Mechanic changes the state of the object Car from "broken" to "fixed" through the process of Repairing using Tool as an instrument. (c) The process Lighting changes the state of the object Lamp from "off" to "on". (d) Suppressing the states of Lamp converts the two effect links yields a single bi-directional effect link.

Two different kinds of links are used in the OPD of Figure 1 to connect objects to processes, depending on the roles that these objects play in the process to which they are linked. Objects may serve as enablers – instruments or intelligent agents, which are involved in the process without changing their state. Objects may also be transformed (change their state, generated, consumed, or affected) as a result of a process acting on them. In this case, the transformation link, denoted by a single- or double-head arrow is used. If a process generates an object, which did not exist prior to the process occurrence, the transformation link is a result link.

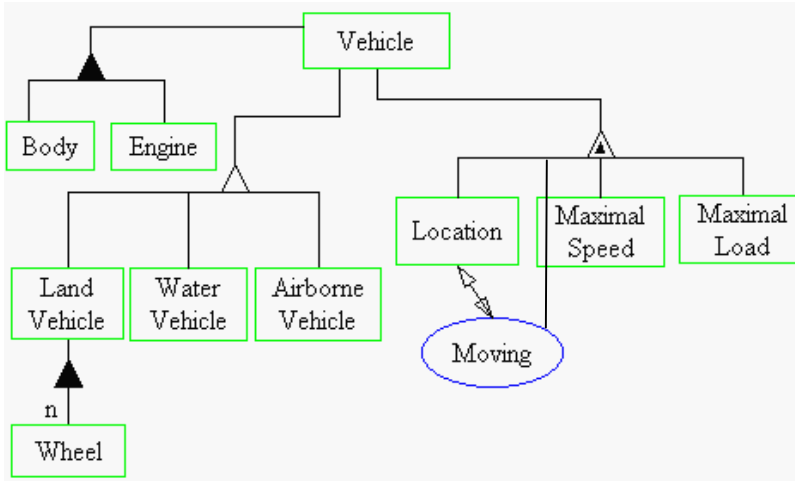

Fig. 2. Compound objects and processes. (a) Vehicle is a compound object, which features (is characterized by) the simple objects Location, Maximal Speed and Maximal Load, which are its attributes, and by the operation Moving. Vehicle specializes into Land Vehicle, Water Vehicle and Airborne Vehicle, and aggregates (has parts, or consist-of) Body and Engine. Land-Vehicle, being a specialization of Vehicle, inherits Vehicle's attributes and parts. Being a Land-Vehicle, it also has n wheels as parts.

Both objects and processes, generalized as "things", can be simple or compound. A compound thing is a thing which has at least one of the following characteristics: It generalizes other things, it aggregates other things pr it features (is characterized by) other things.

Figure 2 shows an example of a compound object. OPM handles the complexity of systems by using recursive seamless scaling. Objects and processes are first presented at a top-level system-environment OPD, which abstracts the key things – objects and processes – that play major roles in the system and its environment. Other, increasingly lower-level OPDs, are zoomed-in views into the details of these compound objects and processes. Selective scaling provides for focusing attention on various system portions at desired detail levels without losing the "big picture".

The time line in an OPD flows from the top of the diagram to its bottom. Hence, the top-to-bottom of process layout in the OPD represents their default execution order. Processes at the same height represent either parallel or mutually exclusive processes, depending on the existence of the logical XOR link between them or lack thereof.

A textual description in a natural-like Object-Process Language (OPL) is automatically extracted from the diagrammatic description. It is used for both customer verification and generation of executable code and database schema. To exemplify this, consider the following example of a generic manufacturing company.

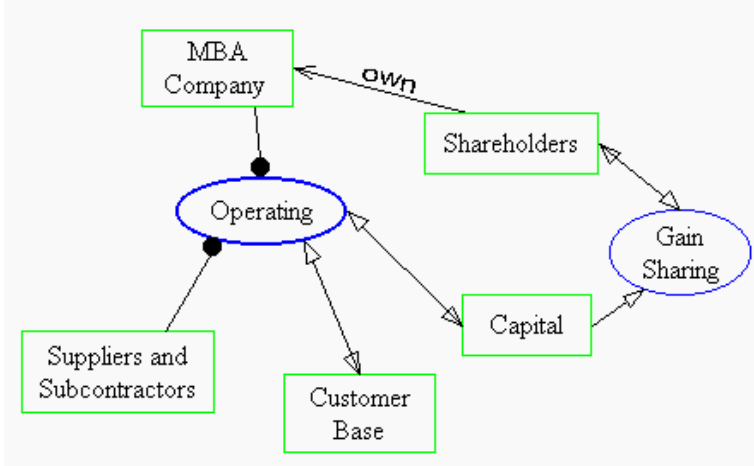

Fig. 3. Top-level Object-Process Diagram of the MBA Company

The Object-Process Language (OPL) set that is equivalent to the OPD of Figure 3 is:

```
  Shareholders own MBA Company.
  Operating is enabled by MBA Company, acting as agent.
  Operating is enabled by Suppliers & Subcontractors,
acting as agent.
  Operating affects Capital and Customer Base.
  Gain Sharing affects Shareholders.
  Gain Sharing consumes Capital.
```

Part of the Object-Process Language (OPL) set that is equivalent to the OPD of Figure 4 is:

```
  Company consists of CEO, CIO, Engineering Department,
Logistics Department, Manufacturing Department, Marketing
Department and Sales Department.
  Managing is enabled by CEO, acting as agent.
  Managing affects MBA Company.
  Operating is enabled by CIO, acting as agent.
  Operating consists of Designing, Manufacturing,
Marketing and Selling.
  Designing is enabled by Engineering Department, acting
as agent.
  Designing is enabled by Marketing Department, acting as
agent.
  Designing consumes capital.
  Designing results in Model.
  Manufacturing is enabled by Model.
  Manufacturing results in Product.
  Manufacturing consumes Capital.
  Marketing is enabled by Product.
  Marketing affects Customer Base.
  Marketing consumes Capital.
  Selling is enabled by Customer Base, acting as agent.
  Selling results Capital.
```

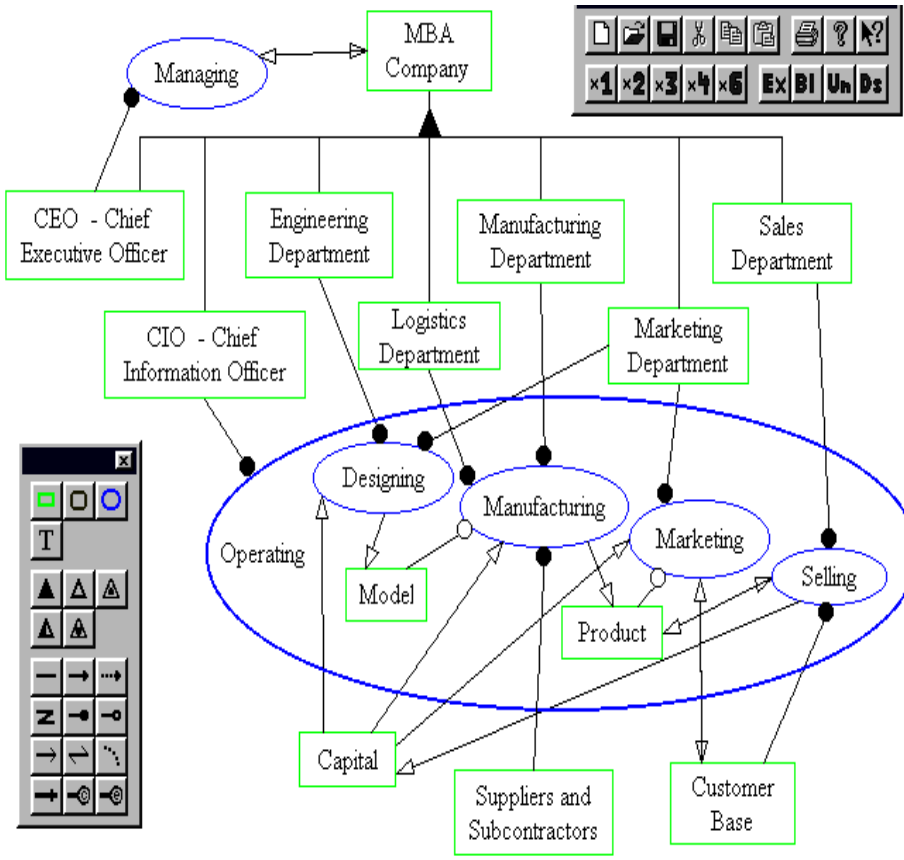

Fig. 4. A zoomed-in Object-Process Diagram, where the object MBA Company is unfolded and the process Operating is inflated.

As the example shows, OPL is designed such that it is very close to English as a natural language, albeit with stringent and limited syntax. The similarity of OPL to the *lingua franca* natural language makes it readable and understandable to humans without the need to learn any programming or pseudo-code-like language. The system's OPL specification resulting from an OPD set is thus amenable to being checked by domain experts, who need not be software experts. Since OPL is close enough to natural English, no prior training is required. This closes the gap between the requirement specification, which the customer usually expresses in prose, and the actual system specification resulting from the OPM analysis and design. On the other hand, the syntax of OPL is well defined and unambiguous. This eliminates the problem of the fuzziness of natural languages and provides a firm basis for executable code generation, database schema definition, control of computer supported collaborative work and other computer based processes.

Thus OPL serves two goals. One is to convert the set of Object-Process Diagrams (OPDs) into a natural-language-like text for use as a means for communicating analysis results back to the prospective users and customers, who may be more comfortable with reading text than checking Object-Process Diagrams.

The other goal of OPL is to provide the infrastructure needed for code generation, database scheme generation and reverse engineering activities.

OPM is useful for understanding system-related problems, communicating with application experts, preparing documentation, and designing solutions for the modeled system, which, as noted, can be of any domain. OPM has been successfully applied in a variety of areas, including studyware design [6] Computer Integrated Manufacturing [7], R&D Management [8] and real-time systems [9]. In the specific area of DAS, OPM was instrumental in works dealing with the Machine Drawing Understanding System (MDUS) [10, 11], 3-D reconstruction from engineering drawings [12] and content-based image retrieval [13].

3 Document Management System – An OPM Representation

Data, information and knowledge are the three major items in the informatics hierarchy. Knowledge, the highest item, is generated by humans from information, which, in turn, is extracted from data. A document is a human-generated artifact that records a representation of some data, information or knowledge. In the long term, interesting documents from the single enterprise or the entire community viewpoint are those that represent knowledge – a meaningful and useful digest of information and data. A document is the instrument that enables preservation of this knowledge, its transfer to other interested parties and extracting the knowledge through capturing the document and analyzing it.

As the OPD in Figure 5 shows, a Document Management System (DMS) consists of three subsystems: generation, exchange and analysis. The Document Generation Subsystem is the instrument used by the document author to generate the document. The Document Exchange Subsystem is the instrument that enables the communication of the document to its prospective audience, which can be a particular human or organization, or the public at large. Exchanging documents implies a two way subsystem: one that publishes (sends or disseminates) the document, and the other that acquires (receives) it. Once acquired, the document must be analyzed, i.e. properly processed, in order to make the knowledge recorded in it in a representational format that is readily available to humans.

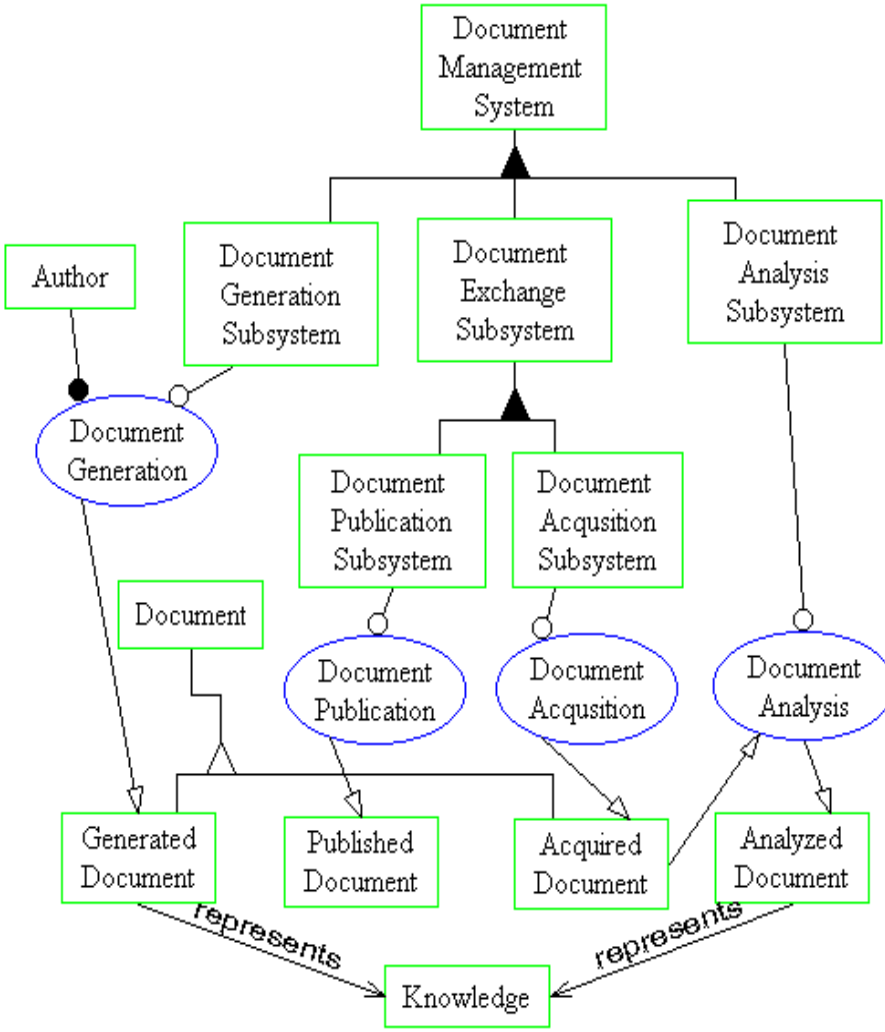

Fig. 5. A top-level OPD of a generic Document Management System showing its three major subsystems and their associated processes.

In the OPD of Figure 5, the two most important Document attributes – Modality and Medium, are specified. Modality has the values analog and digital, while Medium has the values text, image, audio and video. Thus, documents can be recorded in two major different modalities: analog or digital. Documents with analog Modality are paper-based, while those with electronic Modality are electronic-based. As Figure 6 shows, Paper Document can inherit only the text and/or image values of the Medium attribute, while Electronic Document can inherit all the four Medium attribute values.

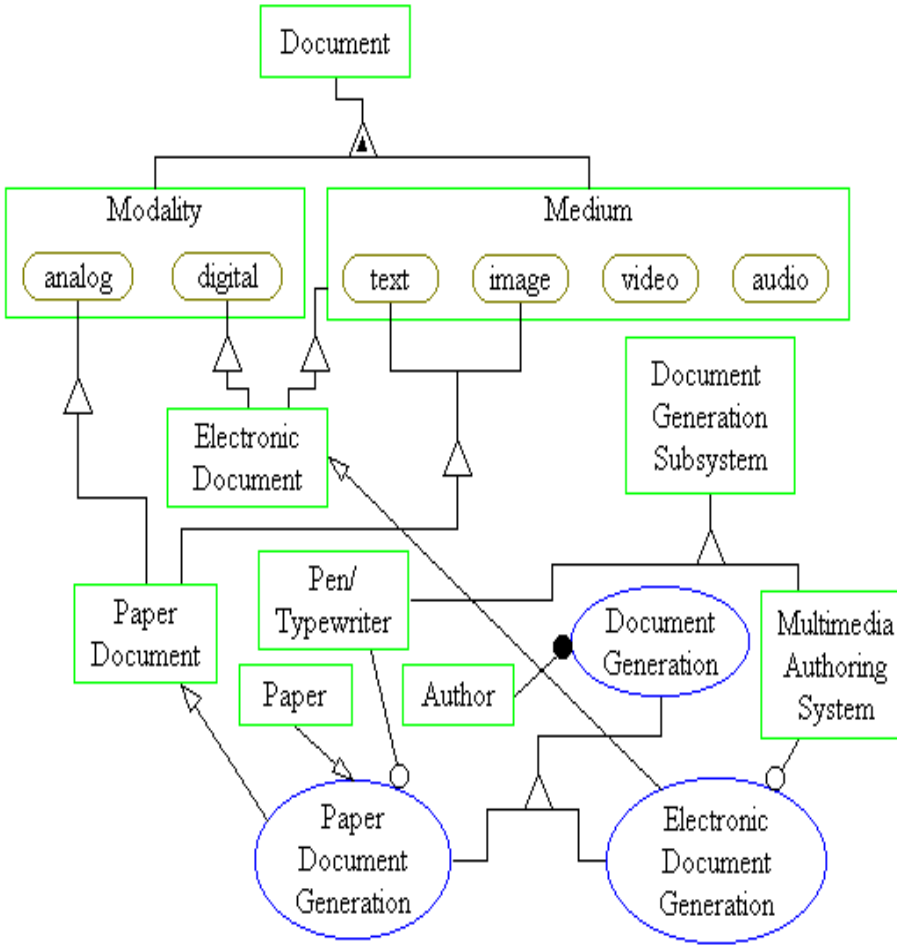

Fig. 6. The attributes of Document and the corresponding specializations of Document Generation and Document Generation Subsystem.

Each modality implies a different implementation of all the three DMS subsystems. Figure 6 shows the attributes of Document and the corresponding specializations of the process Document Generation and the object Document Generation Subsystem. The process Document Generation is enabled by Author, acting as agent. Document Generation Subsystem specializes into Pen/Typewriter, which is the instrument for the Paper Document Generation process, and Multimedia Authoring System, which is the instrument for the Electronic Document Generation process.

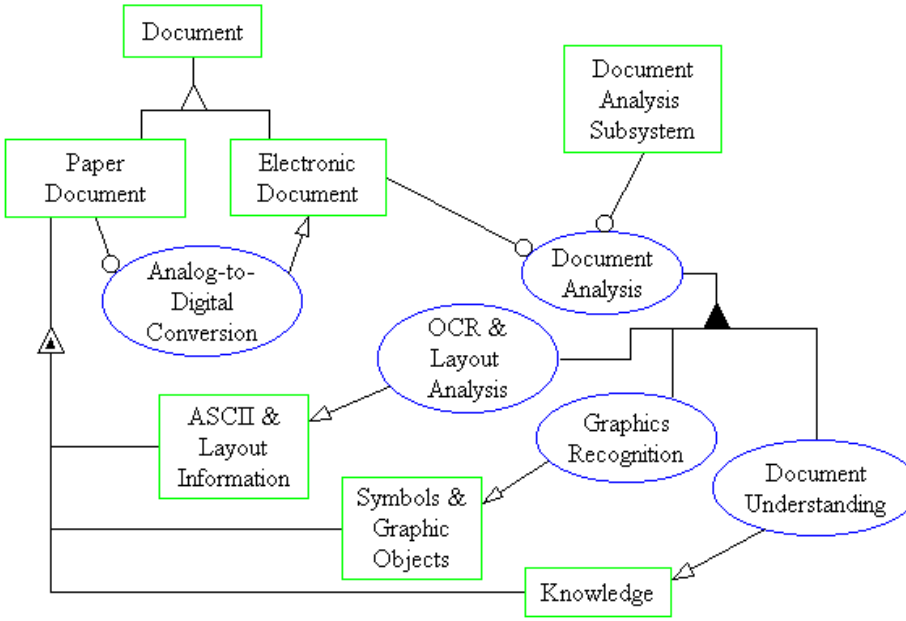

Fig. 7. The Document Analysis Subsystem and the parts of the Document Analysis process.

Figure 7 is an OPD that focuses on the Document Analysis Subsystem. It shows the parts of the Document Analysis process and the attributes of the paper document that are gradually exposed as a result of the various analysis processes. To be analyzed, a Paper Document must first be converted to an Electronic Document. The Analog-to-Digital Conversion process does this. The Electronic Document is input to the Document Analysis process, which consists of three major parts: OCR & Layout Analysis, Graphics Recognition and Document Understanding.

OCR & Layout Analysis (and/or handwriting recognition) yield the textual content of the document information in ASCII form as well as information on the document's layout. Graphics Recognition yields the Document's Symbols & Graphic Objects, while the most sophisticated sub-process – Document Understanding – is aimed at extracting the meaningful knowledge that the original paper document represents in a form that can be best digested by humans. Document Understanding makes use of all the information obtained by the pervious Document Analysis parts and requires a substantial amount of what is known as "artificial intelligence." An example of a high level Document Understanding process is 3-D reconstruction of objects described in engineering drawings. Most contemporary document analysis systems are just beginning to show signs of real document understanding. Indeed, having solved the early analysis tasks, the challenge of current systems is to enhance their high-level understanding capabilities so that they become more intelligent and therefore more useful.

4 Summary

After introducing Knowledge Management and discussing its emergence as a new engineering discipline, we presented the basic principles of the Object-Process Methodology (OPM), its two major tools – Object-Process Diagrams (OPD) – and its complementary textual equivalent – Object-Process Language (OPL). The OPDs were drawn with OPCAT – Object-Process CASE Tool, which was developed to support OPM-based development.

Having argued that documents of interest represent human knowledge, using OPM we then moved to describing a Document Management System as the tool for handling documents. We distinguished three major components of document management: generation, transmission and analysis, and, using an OPD set, we described the details of each component, its structure and behavior.

The work presented in this paper demonstrates the viability of OPM as a standard vehicle for system specification in general and DAS's in particular. The unification of structure and behavior within a single model is a unique feature of OPM that results in synergy founds in no other systems development method.

Acknowledgement

This work is supported by Technion VPR Fund and the Israeli Ministry of Science.

References

1. D. Dori, Object-Process Analysis: Maintaining the Balance Between System Structure and Behavior. Journal of Logic and Computation. 5(2) 227-249, 1995.
2. D. Dori, Unifying System Structure and Behavior through Object-Process Analysis. Journal of Object-Oriented Analysis, July-August, 66-73, 1996.
3. D. Dori and M. Goodman, On Bridging the Analysis-Design and Structure-Behavior Grand Canyons with Object Paradigms. Report on Object Analysis and Design, 2(5), 25-35, January-February 1996.
4. D. Dori and M. Goodman, From Object-Process Analysis to Object-Process Design, Annals of Software Engineering, 2, 20-25, 1996.
5. Booch, G., Jacobson, I., and Rumbaugh, J., Unified Modeling Language (UML) Notation Guide Version 1.1, Rational Software Corporation, September 1 1997.
6. D. Dori and Y.J. Dori, Object-Process Analysis of a Hypertext Organic Chemistry Studyware, Journal of Computers in Mathematics and Science Teaching, 15, 1/2, (1996),65-84.
7. D. Dori, Object-Process Analysis of Computer Integrated Manufacturing Documentation and Inspection Functions. International Journal of Computer Integrated Manufacturing 9(5), 339-353, 1996.

8. D. Meyersdorf and D. Dori, The R&D Universe and Its Feedback Cycles: an Object-Process Analysis R&D Management, 27 (4), 333-344, 1997.
9. M. Peleg and D. Dori, Extending the Object-Process Methodology to Handle Real-Time Systems. Journal of Object-Oriented Programming (to appear).
10. D. Dori, Representing Pattern Recognition Embedded Systems through Object-Process Diagrams: the Case of the Machine Drawing Understanding System Pattern Recognition Letters 16 (4), 377-384, 1995.
11. D. Dori, Arc Segmentation in the Machine Drawing Understanding Environment IEEE Transactions of Pattern Analysis and Machine Intelligence (T-PAMI) 17 (1), 1057-1068, 1995.
12. D. Dori and M. Weiss, A Scheme for 3D Object Reconstruction from Dimensioned Orthographic Views, Engineering Applications in Artificial Intelligence 9 (1), 53-64, 1996.
13. Dov Dori and Hagit Hel-Or: Semantic Content-Based Image Retrieval Using Object-Process Diagrams. In A. Amin, D. Dori, P. Pudil and H. Freeman (Eds.) Advances in Pattern Recognition, Lecture Notes in Computer Science, Vol. 1451, 230-241, 1998.